ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/220267791

An Introduction to Fuzzy State Automata.

Conference Paper · June 1997

DOI: 10.1007/BFb0032485 · Source: DBLP

CITATIONS
27

READS
2,501

1 author:

L.M. Reyneri
Polytechnic University of Turin
134 PUBLICATIONS **1,588** CITATIONS

SEE PROFILE

All content following this page was uploaded by L.M. Reyneri on 21 May 2014.

The user has requested enhancement of the downloaded file.

An Introduction to Fuzzy State Automata

L.M. Reyneri

Dipartimento di Elettronica - Politecnico di Torino
C.so Duca Abruzzi, 24 - 10129 Torino - ITALY
e.mail reyneri@polito.it; phone ++39 11 568 4038; fax ++39 11 568 4099

Abstract

This paper introduces Fuzzy State Automata for control applications. They are derived from the integration of traditional finite state automata and neuro-fuzzy systems, where a finite state automaton tracks the state of the plant under control and modifies the characteristic of the neuro-fuzzy system accordingly. The main difference with respect to existing systems is that the states of the automata are identified by fuzzy, instead of crisp variables, therefore state transitions and the corresponding controller characteristic are smoother and easier to train or tune.

1 Introduction

Fuzzy logic [2] and neural networks [1] are currently used in several control applications [3] due to their interesting performance. They allow either to include "human knowledge" into a controller design, or to design learning controllers which can be trained "by examples", or both of them (neuro-fuzzy integration). Neuro-fuzzy systems are also applied in several other fields, such as pattern classification, function approximation, and many others.

Unfortunately, all fuzzy systems and most neural networks are feed-forward systems, with no feedback, and therefore with no memory. This limits the applicability of neuro-fuzzy systems, especially in the field of control, where most controllers must have memory.

In particular, recurrent neural networks [1] do have memory, but they are difficult to train and they cannot use the human knowledge directly (this is valid for any neural network), unlike fuzzy system which can learn from human experience but which are seldom used in a recurrent architecture, due to the difficulty to deal with feedback in fuzzy logic. Furthermore, feedback in recurrent networks is often limited to simple variables (for instance, speed, acceleration, etc.).

This paper introduces Fuzzy State Automata, a method to design recurrent neuro-fuzzy systems and to deal in a neuro-fuzzy fashion with sequences of states and events. Section 2 first describes how neuro-fuzzy systems can be interfaced with traditional Finite State Automata, and then introduces the theory of Fuzzy State Automata. Section 3 describes their operating principles, while section 4 describes few applications.

Throughout the paper, FFSA are described bearing in mind their application to control systems, but the proposed theory can also be applied to other domains.

2 State of the Art

Many plants may have many different states, according to operating conditions. Examples are: the different strokes of strongly non-linear trajectories, control of robot arms with different load conditions; forward and backward steps of walking robots; optimal operation of tooling machines with different tools; etc.

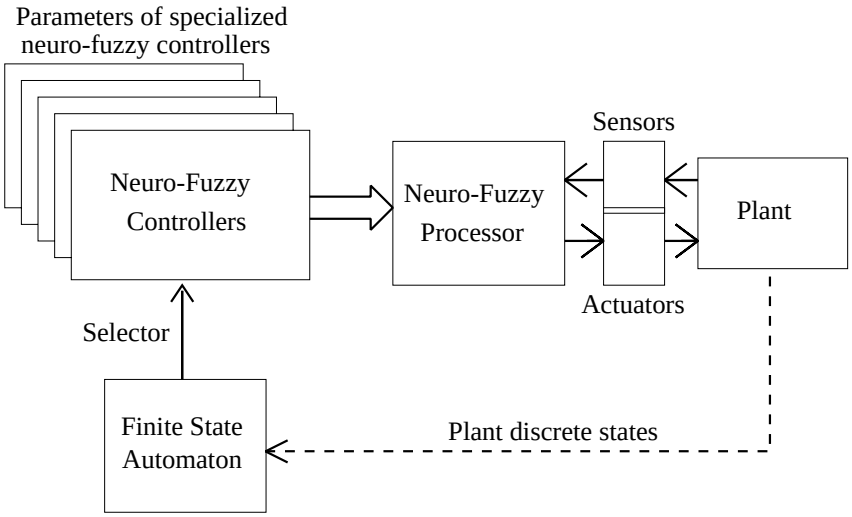

Figure 1: Interaction of neuro-fuzzy controllers with FSA in control applications.

Neuro-fuzzy control methods can be used in conjunction with traditional Finite State Automata (**FSA**) to deal with the different plant states, for the following reasons:

1. fuzzy logic allows to include the available "human knowledge" into the solution of a problem [2], but is not well suited to deal with sequences of events;

2. neural networks may deal with time-dependent signals (with an appropriate training), but they cannot easily acquire the human knowledge;

3. FSA are ideally suited to deal with sequences of events, but they are not good at handling continuous signal and human knowledge;

4. fuzzy systems are mostly feed-forward systems, without feedback, as fuzzy logic is mostly associated with memoryless systems. This is because no method has been found so far to describe efficiently recurrent fuzzy systems;

5. mixing neuro-fuzzy systems with FSA allows to include human knowledge and training capabilities into recurrent systems (therefore, with memory) which can therefore deal with sequence of events, different plant states, etc.

Neuro-fuzzy controllers interacting with FSA have already been used in control applications [5, 6]. The basic idea is sketched in fig. 1: a neuro-fuzzy system controls a plant through a set of sensors and actuators, in a quite well known fashion. A traditional FSA tracks the discrete plant states and modifies the controller characteristic accordingly, by selecting one out of many weight matrices (or knowledge bases, for fuzzy controllers). A different weight matrix (or knowledge base) is associated with each state of the FSA.

Plants operating with a set of well defined states may either be controlled by a single controller, or by a set of simpler controllers interacting with a FSA. The advantages of the second solution are the following:

1. the overall controller is subdivided into a set of simpler controllers; each of these may often be as simple as a linear controller;

2. each controller will be trained only over a limited subset of the state space;

3. each controller has a reduced size and can be implemented in an optimal way, also by using different paradigms for each of them;

4. controllers can be trained independently of each other, therefore training one of them does not affect any of the others.

Unfortunately, traditional FSA interacting with neuro-fuzzy controllers have few major drawbacks:

1. The overall controller characteristic may change abruptly when the FSA has a state transition. This may often cause very high accelerations and jerks which may increase mechanical stresses, vibrations, and often reduce comfort and possibly also the lifetime of the plant.

2. To reduce discontinuities, there is the need for either an additional "smoothing" subsystem (but this often worsens the overall system performance), or the individual controllers should be designed to take care of state changes (but this increases the complexity and the duration of training)

The following sections describe *Fuzzy State Automata* (**FFSA**), which are an alternative solution to the problems listed above. They are derived from the tight integration of neuro-fuzzy systems and more traditional FSA, but they have better performance and are more suited to design smooth characteristics and to deal with sequences of events.

2.1 Fuzzy State Automata

The main difference of FFSA with respect to traditional FSA is that transitions in the automata are not triggered by crisp events but by fuzzy variables, and state transitions are fuzzy as well. It immediately results that, at any time, the whole system is not necessarily in one and only one well-defined state, but it may well be in more states at the same time, each one associated with its own *membership value*.

State transitions are therefore smoother and slower, even if the controllers activated by each state are not designed to smooth the transitions. As a consequence, all the individual controllers may become as simple as a traditional PID, each one designed for a different target specific of that state. The FFSA then takes care of smoothing discontinuities between partial local characteristics, but the overall system is much simpler to design and tune than a traditional controller with similar characteristics.

Unfortunately FFSA have a drawback, that is, as the automata are often in more than one state, they have to process more than one controller at any time. This increase of computing time often counterbalances the higher speed that can be achieved through the use of simpler controllers. In practice, FFSA achieve approximately the same speed of traditional FSA controlling neuro-fuzzy controllers, but they are easier to design (especially for complex systems) and produces smoother trajectories.

So far, the theory of FFSA has been tested on some applications in the field of control, for instance to design the Motion and Leg Coordination Controls of a walking hexapod [6], where FFSA have been used as a way to specify the cooperation between individual legs. This example is shortly described in section 4.

3 Operation of Fuzzy State Automata

This section describes theory and operation of FFSA, and in particular how FFSA can be converted to fuzzy descriptions and the corresponding mathematical models. The theory will be described by means of the example shown in fig. 2. This is nothing but a simple example with only one input variable, but the diagram could well be part of a more complex FFSA. Furthermore, the theory applies also to FFSA with more than one input.

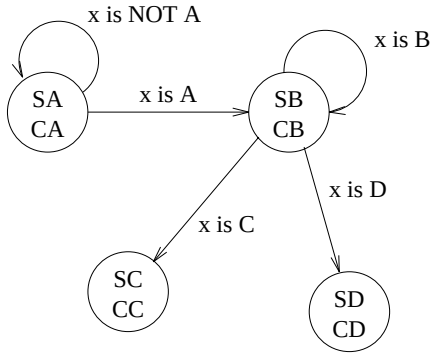

Figure 2: An example of an FFSA.

An FFSA looks very similar to a traditional FSA, in the sense that it can be represented as a collection of *fuzzy states* S_j (namely, the circles), connected by *fuzzy transitions* (namely, the arrows). Each fuzzy transition is labeled by a fuzzy expression which can be as complex as desired (A, B, C, D are traditional *fuzzy sets*).

As in a traditional FSA, a state represents univocally the operating conditions of an FFSA, at any given instant. But, unlike FSA, a system need not be in only one state at a time. Each state S_j is therefore associated with a *fuzzy state activity* $\mu_{S_j} \in [0, 1]$, which represents "how much" the system is in that particular state. The state activity is somehow equivalent to the degree of membership of a fuzzy variable, yet note that the state in an FFSA is not a fuzzy variable.

If we associated a state activity to states in a traditional FSA, it would be:

$$\mu_{S_j} = \begin{cases} 1, & \text{for the active state} \\ 0, & \text{for all other states} \end{cases} \tag{1}$$

with the constraint that the *total activity*:

$$\sum_j \mu_{S_j} = 1, \tag{2}$$

meaning that the system is always in one and only one well defined state S_j (the one with $\mu_{S_j} = 1$) and in no other state.

Constraint (2) applies also to FFSA, therefore activity can be distributed among several states and can partially move from one state to another (possibly more than one), but the total activity shall always be constant and equal to 1.

FFSA can be of different types: *time-independent* and *time-dependent*; the former can either be *synchronous* or *asynchronous*, which somewhat reflect synchronous and asynchronous FSA, respectively. They differ only in the way state activity moves from one state to another, according to the degrees of membership of the fuzzy transitions. Roughly we could say that:

1. in asynchronous, time-independent FFSA, state transitions may take place as soon as inputs vary (yet transitions are not so abrupt as in traditional FSA);

2. in synchronous, time-independent FFSA, state transitions are computed in a way similar to asynchronous FFSA, but they are applied only at the next clock cycle;

3. in time-dependent FFSA, there is always an intrinsic delay (usually larger than the clock period, if any) between input variations and the corresponding state transitions.

3.1 Time-independent automata

In time-independent FFSA, the activity moves from each state to one or more other states, as a function of the present state activity and the degrees of membership of the state transitions, but independently of time (hence the name). State activity may "move" only along state transitions.

In more details, a time-independent FFSA (either synchronous or asynchronous) can be translated into a more traditional fuzzy description made of one rule per each transition, where each rule is in the form:

$$\text{IF (STATE is oldstate) AND (fuzzyexpression) THEN (STATE is newstate)} \qquad (3)$$

where the term (STATE is S_j) has, by definition, a degree of membership equal to the state activity μ_{S_j}, whereas the AND, OR and NOT implicators have the traditional meaning of any fuzzy system [2]. Two rules with the same consequent are supposed to be connected by an OR implicator.

Transitions starting from and ending into the same state must also be taken into account. For instance, for the example shown in fig. 2, the equivalent time-independent fuzzy description is made of five rules:

$$\begin{aligned}
&\text{IF (STATE is SA) AND (x is NOT A) THEN (STATE is SA)}\\
&\text{IF (STATE is SA) AND (x is A) THEN (STATE is SB)}\\
&\text{IF (STATE is SB) AND (x is B) THEN (STATE is SB)}\\
&\text{IF (STATE is SB) AND (x is C) THEN (STATE is SC)}\\
&\text{IF (STATE is SB) AND (x is D) THEN (STATE is SD)}
\end{aligned} \qquad (4)$$

A constraint has to be placed on the membership functions $\mu_i(x)$ associated with all the transitions exiting from a state, to let the FFSA operate properly (not proven here):

$$\sum_{i \in \mathcal{T}_j} \mu_i(x) \approx 1 \qquad \forall x \in \mathcal{U}_x, \quad \forall j \qquad (5)$$

where $\mathcal{T}_j$ is the set of transitions exiting from state S_j and $\mathcal{U}_x$ is the *universe of discourse* of the input variable x. For the example of fig. 2, the above identity holds if and only if $B =$ NOT (C OR D).

In traditional FSA, the transitions exiting from a state must be both *exhaustive* (meaning that the logic sum of all transition labels must always be true) and *mutually exclusive* (meaning that no two transitions must be active at the same time). Similarly, in FFSA, constraint (5) is equivalent to exhaustivity, but there is no equivalent of mutual exclusion.

As any other fuzzy system, the fuzzy equivalent of an FFSA also has a mathematical model. Supposing to use the min and max operators as AND and OR implicators, respectively, the fuzzy system (4) results into the following mathematical expression, valid only for synchronous FFSA (one equation per state):

$$\begin{aligned}
\mu'_{\mathrm{SA}}(t+\Delta t) &= \max\{\min\{\mu_{\mathrm{SA}}(t), 1-\mu_{\mathrm{A}}(x)\},\ \ldots\}\\
\mu'_{\mathrm{SB}}(t+\Delta t) &= \max\{\min\{\mu_{\mathrm{SA}}(t), \mu_{\mathrm{A}}(x)\},\ \min\{\mu_{\mathrm{SB}}(t), \mu_{\mathrm{B}}(x)\}\ \ldots\}\\
\mu'_{\mathrm{SC}}(t+\Delta t) &= \max\{\min\{\mu_{\mathrm{SB}}(t), \mu_{\mathrm{C}}(x)\},\ \ldots\}\\
\mu'_{\mathrm{SD}}(t+\Delta t) &= \max\{\min\{\mu_{\mathrm{SB}}(t), \mu_{\mathrm{D}}(x)\},\ \ldots\}
\end{aligned} \qquad (6)$$

where the dots stand for any other transition reaching the corresponding states (not shown in fig. 2). The values $\mu'_{S_j}(t+\Delta t)$ represent the *new state activities* at time $(t+\Delta t)$, where Δt is the sampling period.

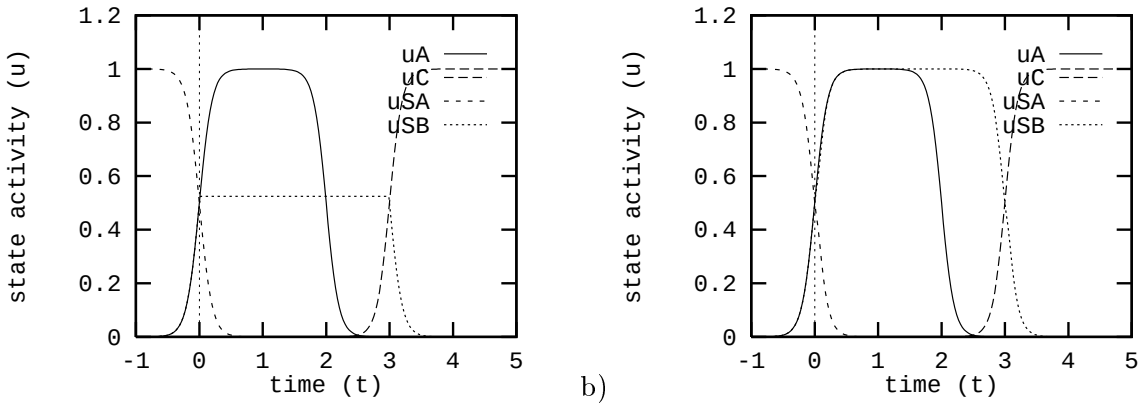

Figure 3: a) a deadlock in an FFSA; b) correct behavior of an asynchronous time-independent FFSA. Simulation results with $\mu_{\mathrm{D}}(x) \approx 0$ and $\mu_{\mathrm{B}}(x) \approx (1 - \mu_{\mathrm{C}}(x))$

This might seem a straightforward implementation of an FFSA, but it does not guarantee that total activity remains constant. In practice, it suffers from the problem depicted in fig. 3.a: suppose that, at $t = 0$, the FFSA is mostly in state SA (namely, $\mu_{\mathrm{SA}} \approx 1$, $\mu_{\mathrm{SB}}, \mu_{\mathrm{SC}}, \mu_{\mathrm{SD}} \approx 0$), $\mu_{\mathrm{A}}(x), \mu_{\mathrm{C}}(x), \mu_{\mathrm{D}}(x) \approx 0$, therefore $\mu_{\mathrm{B}}(x) \approx 1$ (from (5)). In this case equations (6) simplify and, as soon as $\mu_{\mathrm{A}}(x)$ begins to monotonically increase:

$$\begin{aligned} \mu'_{\mathrm{SA}}(t+\Delta t) &= \min\{\mu_{\mathrm{SA}}(t), (1-\mu_{\mathrm{A}}(x))\} \;=\; (1-\mu_{\mathrm{A}}(x)) \\ \mu'_{\mathrm{SB}}(t+\Delta t) &= \max\{\min\{\mu_{\mathrm{SA}}(t), \mu_{\mathrm{A}}(x)\},\ \mu_{\mathrm{SB}}(t)\} = \\ &= \max\{\min\{(1-\mu_{\mathrm{A}}(x)), \mu_{\mathrm{A}}(x)\}, \mu_{\mathrm{SB}}(t)\} \end{aligned} \tag{7}$$

Therefore, as long as $\mu_{\mathrm{A}}(x) \leq 0.5$, the more $\mu_{\mathrm{A}}(x)$ increases, the more the activity moves from SA to SB; in particular, the activity of SB increases and tracks $\mu_{\mathrm{A}}(x)$, and that of SA decreases accordingly, thus total activity remains constant.

The problems arises when $\mu_{\mathrm{A}}(x) > 0.5$. From that point, activity of SA keeps decreasing but that of SB stops and gets stuck at a value of 0.5 (from (7)), as shown in fig. 3.a.

This problem is solved by using constraint (2) to normalize the state activities given by equation (6), therefore:

$$\mu_{\mathrm{S_j}}(t+\Delta t) = \frac{\mu'_{\mathrm{S_j}}(t+\Delta t)}{\sum_k \mu'_{\mathrm{S_k}}(t+\Delta t)} \tag{8}$$

By doing so, the total activity is guaranteed to remains equal to 1.

Instead, asynchronous time-independent FFSA use an alternative criterion to update $\mu_{\mathrm{S_j}}$; the mathematical model of asynchronous FFSA can be obtained by modifying formulae (6) and (8) as, respectively:

$$\begin{cases} \mu'_{\mathrm{S_j}}(t) = f_j\left(\ldots \mu_{\mathrm{i}}(x) \ldots; \ldots \mu_{\mathrm{S_k}}(t) \ldots\right) \\ \mu_{\mathrm{S_j}}(t) = \frac{\mu'_{\mathrm{S_j}}(t)}{\sum_k \mu'_{\mathrm{S_k}}(t)} \end{cases} \tag{9}$$

where $f_j(\cdot)$ are appropriate functions obtained by analyzing the state diagram (for instance, formulae (6)).

The difference with respect to the synchronous FFSA is that formulae (9) give the value of $\mu_{\mathrm{S_j}}$ at time t (and not $(t+\Delta t)$), therefore they are a set of non-linear equations in the variables $\mu_{\mathrm{S_j}}$ (which appear in both sides of (9)) and must be solved appropriately (for instance, by means of relaxation methods).

Also in this case, total activity remains equal to 1; see fig. 3.b, which shows how the activity of SB keeps tracking $\mu_{\mathrm{A}}(x)$ also when larger than 0.5.

The difference between synchronous and asynchronous FFSA is that the the latter is more complex to compute, but the activity of SB tracks perfectly $\mu_{\rm A}(x)$ for any value of it, while the former is faster to compute, but the activity of SB tracks $\mu_{\rm A}(x) > 0.5$ with a slight time delay.

3.2 Time-dependent automata

Like in time-independent FFSA, the activity moves from each state to one or more other states, as a function of the state activity and the degrees of membership of state transitions. But, unlikely time-independent FFSA, variations are slower and depend on time (hence the name). State activity "moves" only along transitions.

In more details, a time-dependent FFSA can be translated into a more traditional fuzzy description made of one rule per each state transition, where each rule is in the form:

$$\begin{array}{l} \text{IF (STATE is oldstate)} \quad \text{AND (fuzzyexpression) THEN} \\ \qquad\qquad\qquad\qquad \text{(STATE flows to newstate)} \end{array} \tag{10}$$

where all terms have the same meaning described in section 3.1, except for the new operator (STATE flows to newstate), whose mathematical model is expressed by two differential equations:

$$\begin{aligned} \frac{d}{dt}\,\mu_{\rm newstate}(t) &= \frac{1}{\tau}\,\mu_{\rm fuzzyexpression} \\ \frac{d}{dt}\,\mu_{\rm oldstate}(t) &= -\frac{d}{dt}\,\mu_{\rm newstate}(t) \end{aligned} \tag{11}$$

For instance, for the example shown in fig. 2, the equivalent time-dependent fuzzy description is:

$$\begin{array}{l} \text{IF (STATE is SA) AND (x is NOT A) THEN (STATE flows to SA)} \\ \text{IF (STATE is SA) AND (x is A) THEN (STATE flows to SB)} \\ \text{IF (STATE is SB) AND (x is B) THEN (STATE flows to SB)} \\ \text{IF (STATE is SB) AND (x is C) THEN (STATE flows to SC)} \\ \text{IF (STATE is SB) AND (x is D) THEN (STATE flows to SD)} \end{array} \tag{12}$$

which produces to the following mathematical model (two equations per rule):

$$\frac{d}{dt}\,\mu_{\rm SA}(t) = \frac{1}{\tau}\min\{\mu_{\rm SA}(t), 1-\mu_{\rm A}(x)\} \quad ; \quad \frac{d}{dt}\,\mu_{\rm SA}(t) = -\frac{d}{dt}\,\mu_{\rm SA}(t) \tag{13}$$

$$\frac{d}{dt}\,\mu_{\rm SB}(t) = \frac{1}{\tau}\min\{\mu_{\rm SA}(t), \mu_{\rm A}(x)\} \quad ; \quad \frac{d}{dt}\,\mu_{\rm SA}(t) = -\frac{d}{dt}\,\mu_{\rm SB}(t) \tag{14}$$

$$\frac{d}{dt}\,\mu_{\rm SB}(t) = \frac{1}{\tau}\min\{\mu_{\rm SB}(t), \mu_{\rm B}(x)\} \quad ; \quad \frac{d}{dt}\,\mu_{\rm SB}(t) = -\frac{d}{dt}\,\mu_{\rm SB}(t) \tag{15}$$

$$\frac{d}{dt}\,\mu_{\rm SC}(t) = \frac{1}{\tau}\min\{\mu_{\rm SB}(t), \mu_{\rm C}(x)\} \quad ; \quad \frac{d}{dt}\,\mu_{\rm SB}(t) = -\frac{d}{dt}\,\mu_{\rm SC}(t) \tag{16}$$

$$\frac{d}{dt}\,\mu_{\rm SD}(t) = \frac{1}{\tau}\min\{\mu_{\rm SB}(t), \mu_{\rm D}(x)\} \quad ; \quad \frac{d}{dt}\,\mu_{\rm SB}(t) = -\frac{d}{dt}\,\mu_{\rm SD}(t) \tag{17}$$

$$\tag{18}$$

where all terms $\frac{d}{dt}\,\mu_{\rm S_j}$ with the same j, at the left sides of the above set of equations must be summed up together. It is clear from (13) and (15) that, unlikely time-independent FFSA, transitions starting from and ending into the same state have no effect at all, as they produce two opposite terms which cancel each other out. Therefore such transitions need not be taken into account.

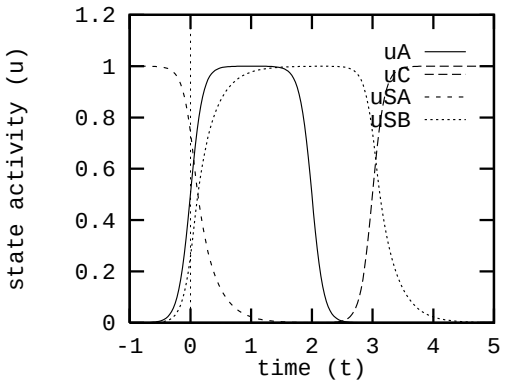

Figure 4: Behavior of a time-dependent FFSA.

The main difference with respect to time-independent FFSA is that in time-dependent FFSA, state activity moves from one state to another with a time constant τ (which may also be different for each transition), therefore there is a time-lag between variations of inputs and of state activities, as shown in fig. 4.

3.3 Integration with neuro-fuzzy systems

So far, FFSA alone have described in details; this section explains how they can be integrated with neuro-fuzzy systems to produce operational controllers to be used in practical applications. As for FFSA, the technique proposed below can also be applied to other domains.

To integrate FFSA with neuro-fuzzy systems(or traditional controllers, as well), it is necessary to associate each state S_j with one controller C_j, as shown in the example of fig. 2.

Controllers can be either linear, non-linear, neural, fuzzy, neuro-fuzzy, look-up tables, or any combination thereof. Furthermore, each controller can be implemented using the technique which best suits the requirements of that particular state, therefore the individual controllers need not be of the same type.

The only constraint is that all controllers must be functions of the same input variables and must produce the same output variables. In vector form:

$$\vec{Y}^{\,j} = C_j\left(\vec{X}; t\right) \tag{19}$$

where $\vec{X}$ and $\vec{Y}$ are the input and output vector, respectively. The overall characteristic, resulting from the integration of the controllers with the FFSA becomes:

$$\vec{Y} = \sum_j \mu_{\mathrm{S_j}}(t) \cdot C_j\left(\vec{X}; t\right) \tag{20}$$

As state activity $\mu_{\mathrm{S_j}}(t)$ moves smoothly from one state to another, when input conditions vary, the overall controller characteristic varies smoothly as well, removing all abrupt changes and variations. This has a lot of benefits in control applications, as described in section 1.

4 Application Examples

This section describes shortly two application examples of FFSA in the field of control, and in particular to control the trajectory of two robots.

The first one is a simple method to define circular (or elliptical) trajectories, by means of an FFSA. Fig. 5.a shows an FFSA with four states and two inputs x and y, which are the

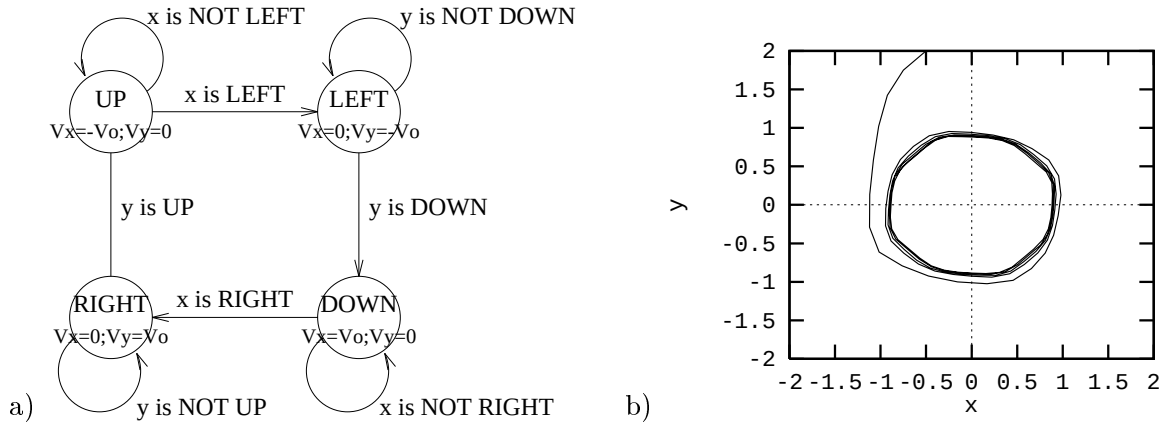

Figure 5: Circular trajectory controlled by a simple FFSA. The trajectory shown in fig. 6.b looks more octagonal than circular; this is due to sampling time in simulations.

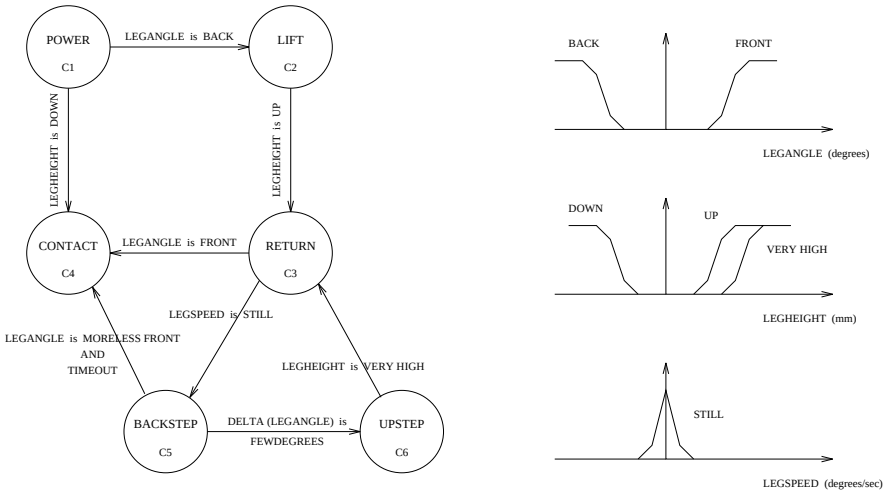

Figure 6: FFSA of the Leg Control.

coordinates of a moving spot (for instance, the robot's tip). Each fuzzy state activates one of four simple controllers, which move the spot towards either left, right, up, or down, at a constant speed. The four fuzzy sets LEFT, RIGHT, UP and DOWN are self-explanatory.

Fig. 5.b shows the resulting trajectory, starting from a point external to the desired trajectory. The speed of the moving spot is almost constant and equal to V_o (the target speed of the four controllers), while the diameter depends on the membership functions LEFT, RIGHT, UP and DOWN. When the four fuzzy sets are symmetrical with respect to axis origin, the trajectory is circular, but it may easily become elliptical, or its center can be moved, by shifting the membership functions. Yet, spot speed remains constant.

Another application is to define the Motion and Leg Coordination Controls of a six-legged robot [6]. This is a quite tough task, which requires a tight interaction between different controllers. A much simpler approach can be obtained using the FFSA shown in fig. 6, which tracks the state of each leg of the robot (for instance, forward step, backward step, leg lift and contact phases). Each state is associated with a different controller, each one tailored to the specific system state. For instance, controllers C1, C2, C3 and C4 are simple PID controllers tuned to, respectively: hold body weight and move the leg backwards; rise the leg at maximum speed; return the leg forward at maximum speed; lower the leg at reduced speed.

The FFSA of fig 6 has provisions for the normal leg step (namely, walking on smooth surfaces

without obstacles) and to recover from obstacles encountered during leg movements. The actual FFSA is more complex, as it includes provisions for: i) modification of hexapod attitude in presence of complex obstacles (e.g. to walk inside low tunnels), ii) modification of leg trajectory when walking sideways, iii) control of alternative paces for dynamic gaits (e.g. galloping, or climbing staircases); iv) other special cases a leg might have to face.

5 Conclusion

The paper has introduced Fuzzy State Automata, as a method which allows a tight integration between finite state automata and neuro-fuzzy systems. Fuzzy State Automata can be converted to more traditional fuzzy systems, therefore they can run on either traditional fuzzy processors, or on systems with fuzzy computing capabilities. Fuzzy State Automata are also a method to describe recurrent fuzzy systems. Two examples have been described in the field of control.

References

[1] S. Haykin, "Neural Networks: A Comprehensive Foundation", *Mc Millan College Publishing Company*, New York, 1994.

[2] L. Wang, "Adaptive Fuzzy Systems and Control", *Prentice Hall*, Englewood Cliffs, New Jersey, 1994.

[3] D.A. White and D.A. Sofge, "Handbook of Intelligent Control", Van Nostrand Reinhold, 1992.

[4] L.M. Reyneri, "Weighted Radial Basis Functions for Improved Pattern Recognition and Signal Processing", submitted to *Neural Letters*.

[5] L.M. Reyneri, M. Chiaberge, L. Zocca, "CINTIA: A Neuro-Fuzzy Real Time Controller for Low Power Embedded Systems", *IEEE MICRO, special issue on Hardware for Artificial Neural Networks*, June 1995, pp. 40-47.

[6] F. Berardi, M. Chiaberge, E. Miranda, L.M. Reyneri, "A Walking Hexapod Controlled by a Neuro-Fuzzy System", Proc. of MCPA'97, Pisa (I), February 1997, pp. 95-104.

View publication stats